



Unit 3

Designing User Interface-2

Lesson 3.1-3

TreeView Control

A TreeView control is designed to present a list in a hierarchical structure. It is similar to a directory listing. Users can open individual nodes that can in turn contain child nodes. The TreeView control is suitable for displaying XML data, but can be used for any data that can be represented in a hierarchy.

Upon completion of this unit you will be able to:



Outcomes

- Create TreeView control.
- Use checkboxes in TreeView.

TreeView Control

In visual studio 2008 a TreeView control is used to present a list or data in a hierarchical structure. Individual list or data name is called node. So you can use a TreeView control to display a hierarchy of nodes. Each node also can have child nodes. In TreeView control have two types of nodes called root node and child node respectively. Child nodes always stay under root nodes. A TreeView also can be displayed with checkboxes next to the nodes. The main properties of TreeView's are Nodes and SelectedNode. The Nodes property contains the list of nodes in the TreeView and the SelectedNode property gets or sets the currently selected node. You can add, remove and clone a TreeNode. You can set the text for each tree node label by setting a TreeNode object's text property. From TreeView's the user can expand a node for showing its child nodes by clicking the plus sign (+) and collapse a node for hiding its child nodes by clicking the minus sign (-).

The public properties of TreeView object are given in the following table:



Sl. No.	Property Name	Description
1.	BorderStyle	This is used to gets/sets the TreeView border style.
2.	CheckBoxes	This is used to gets/sets whether checkboxes should be displayed next to tree nodes.
3.	FullRowSelect	This is used to gets/sets whether a selection should select the whole width of the TreeView.
4.	HideSelection	This is used to gets/sets whether the selected tree node stays highlighted, when the tree view loses the focus.
5.	HotTracking	This is used to gets/sets whether a tree node label should change its appearance when the mouse pointer moves over it.
6.	ImageIndex	This is used to gets/sets the image list index of the current image.
7.	ImageList	This is used to gets/sets the image list used with this TreeView.
8.	Indent	This is used to gets/sets the distance that each level should be indented.
9.	ItemHeight	This is used to gets/sets the height of the tree nodes.
10.	LabelEdit	This is used to gets/sets whether tree node text can be edited.
11.	Nodes	This is used to gets the collection of tree nodes.
12.	PathSeperator	This is used to gets/sets the string the tree node uses as a path delimiter.
13.	Scrollable	This is used to gets/sets whether the tree view should display scroll bars as needed.
14.	SelectedImage Index	This is used to gets/sets the image index for the image to display when a node is selected.
15.	SelectedNode	This is used to gets/sets the node that is selected.
16.	ShowLines	This is used to gets/sets whether the lines are drawn between tree nodes.
17.	ShowPlusMinus	This is used to gets/sets whether plus sign (+) and minus sign (-) buttons are shown next to tree nodes with child tree nodes.
18.	ShowRootLines	This is used to gets/sets whether lines should be drawn between the tree nodes and the root node.
19.	Sorted	This is used to gets/sets if the tree nodes should be sorted.
20.	TopNode	This is used to gets/sets the first visible tree node.
21.	VisibleCount	This is used to gets/sets the number of nodes that can be seen currently.




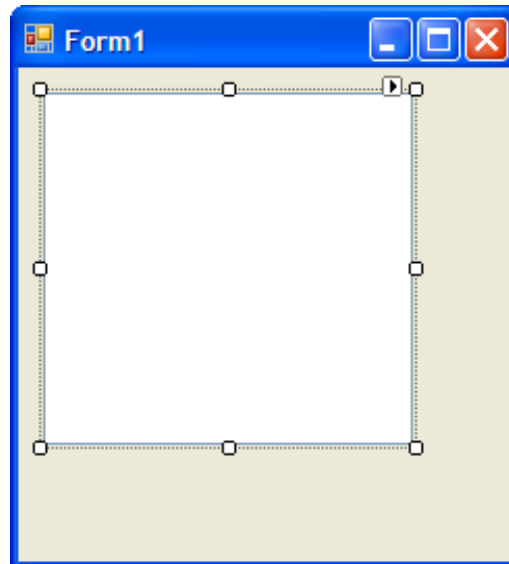
The public methods of TreeView objects are given in the following table:

Sl. No.	Method's Name	Description
1.	BeginUpdate	This is used to disable redrawing of the TreeView.
2.	CollapseAll	This is used to collapse all nodes.
3.	EndUpdate	This is used to enables redrawing of the tree view.
4.	ExpandAll	This is used to expand all the nodes.
5.	GetNodeAt	This is used to gets the node that is at the given location.
6.	GetNodeCount	This is used to gets the number of nodes.

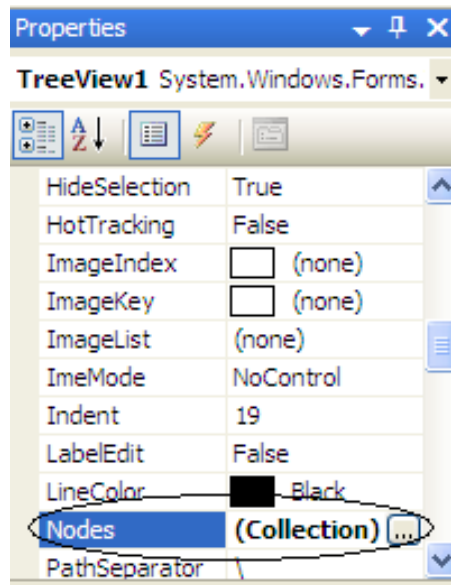
Creating TreeView at Design Time

To create a TreeView at design time follow the following steps:

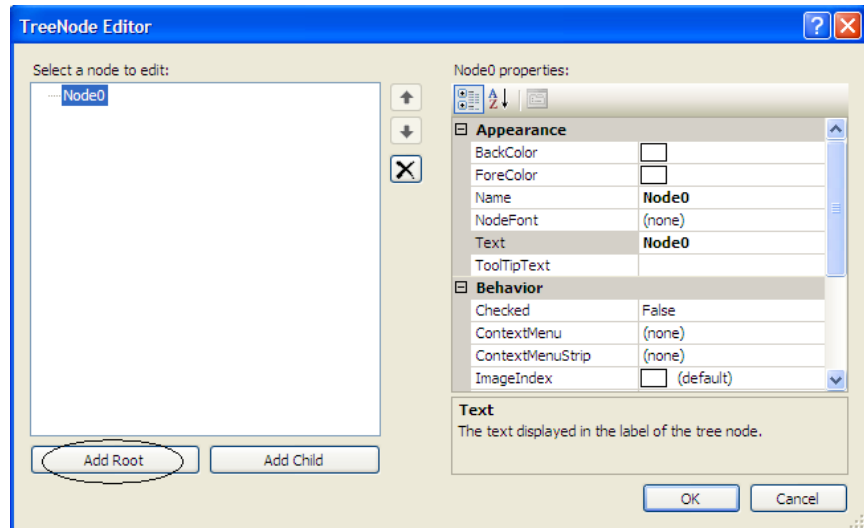
Step 1: Drag a TreeView control  from the toolbox onto a windows form, which will look like the following:



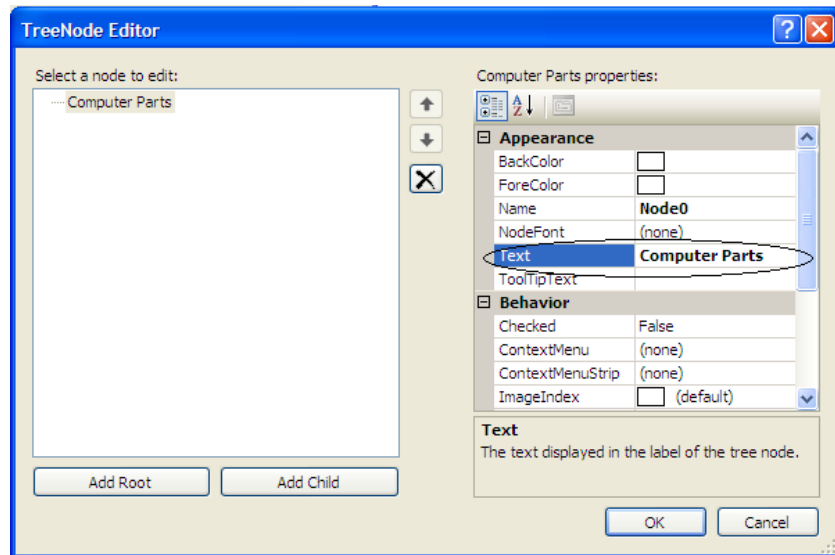
Step 2: Now click on collection option of Nodes property from the properties window, which will look like the following:



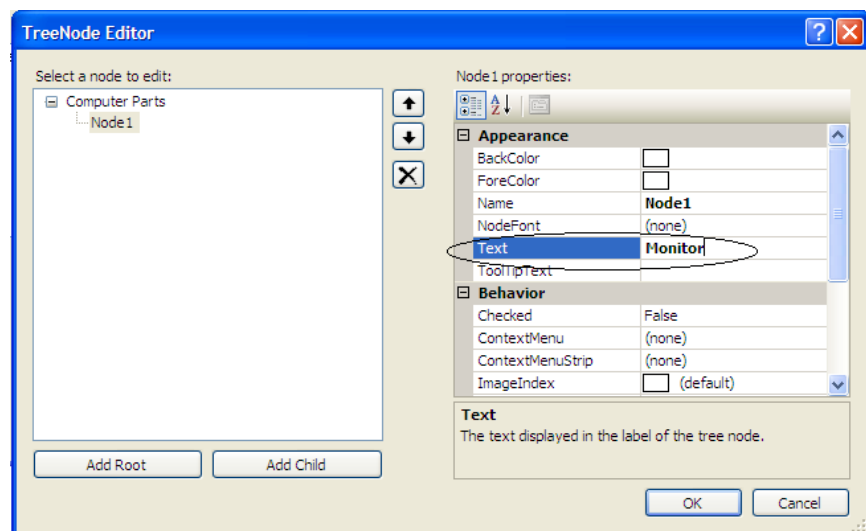
Step 3: Now a TreeNode editor window will appear, and click on Add Root button, which will look like the following:



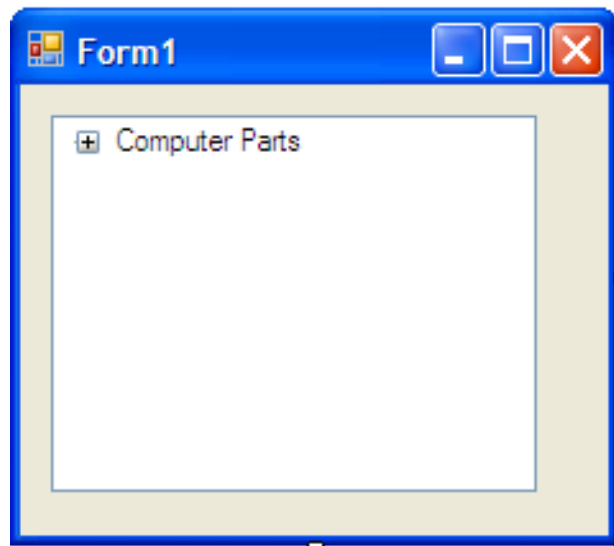
Step 4: Now change the text of Node0 option from the appearance properties and type Computer Parts and then click ok button as like following figure:



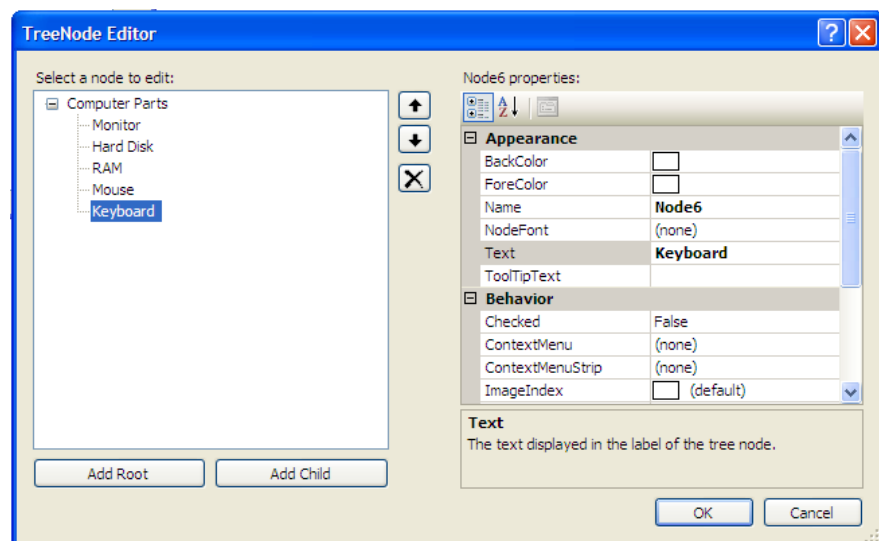
Step 5: According to step 4 you can be create more than one root nodes in TreeView by clicking Add Root button. Now we create child node under tree node. So now click on Add Child button. When you click on Add Child button, by default Computer Parts (which is root node) node is selected, so child nodes are created under this root nodes. Now when you click on Add child button, Node1 option will see under Computer parts node. Now select Node 1 option and then change the text of Node 1 from the appearance properties and type monitor on textbox then click ok as like the following:



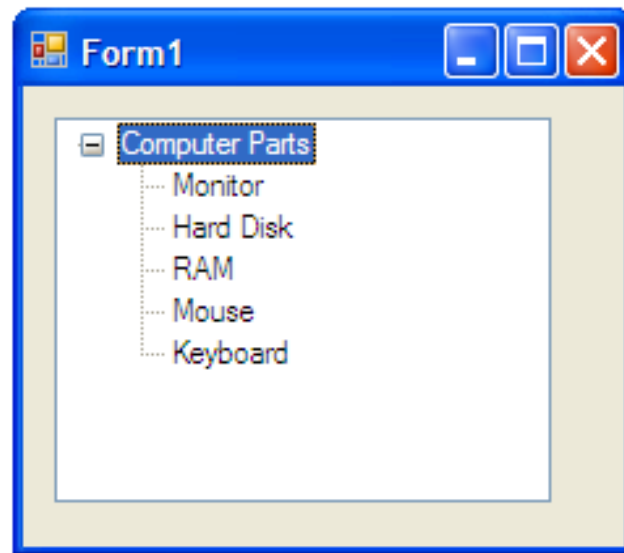
Step 6: After clicking ok button you will look the following form. Now click on + sign, then you will look the Parts Name Monitor under root node Computer Parts.



Step 7: If you want to create more than one node under root node, you should follow the step 6. Here we will create another node under Computer Parts node like Hard disk, RAM, Mouse, Keyboard. So now you select TreeView Control from the windows form and then click collection option of Nodes from the properties window. And then select Computer Parts node and click + sign, then click Add Child button. Here you will see Node 2 is created under Monitor node, then select Node 2 option and change the text from the appearance properties and click ok button. According this procedure you can be creating more than one node under root node. Here we create another four child node as like the following:



Step 8: Now run the program and click + sign of the TreeView control, you will look like the following window:



Handling TreeView Events

TreeView events handling is very simple. TreeView controls have a number of events. Most of the events are shown in Table 1. The default event is the **AfterSelect** event, which occurs after a node has been selected. It is an event of the TreeView control not of the **TreeNode** object that was selected, but you can determine which node was selected with **TreeViewEventArgs** object that is passed to you.

Table 1: Public Events Of TreeView object.

SL.No	Events Name	Description
1.	AfterCheck	This is occurs when a node checkbox is checked.
2.	AfterSelect	This is occurs when a tree node is selected.
3.	AfterCollapse	This is occurs when a tree node is collapsed.
4.	AfterExpand	This is occurs when a tree node is expand.
5.	BeforeCheck	This is occurs before a node checkbox is checked.
6.	BeforeSelect	This is occurs before a node is selected.
7.	BeforeCollapse	This is occurs before a node is collapsed.
8.	BeforeExpand	This is occurs before a node is expand.

**Example:**

Step 1: Consider the example. Now drag and drop a textbox control on windows form.

Step 2: Now just double click on TreeView Control and type the following code under Treeview Control:

```
TextBox1.Text = "you clicked: " & e.Node.Text
```

Full code will be as follows:

```
Public Class Form1

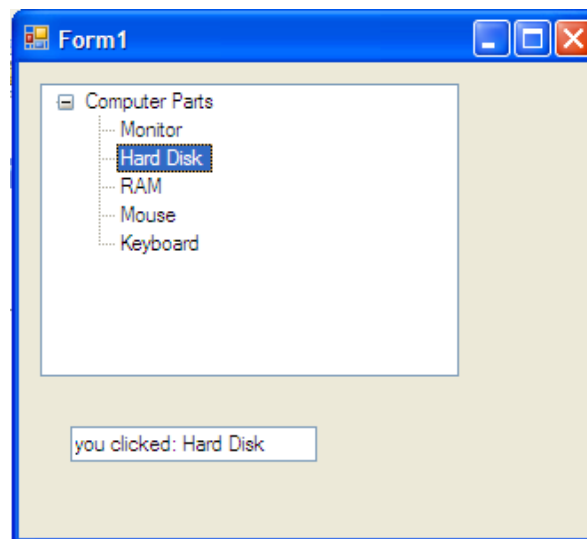
    Private Sub TreeView1_AfterSelect(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles
TreeView1.AfterSelect

        TextBox1.Text = "you clicked: " & e.Node.Text

    End Sub

End Class
```

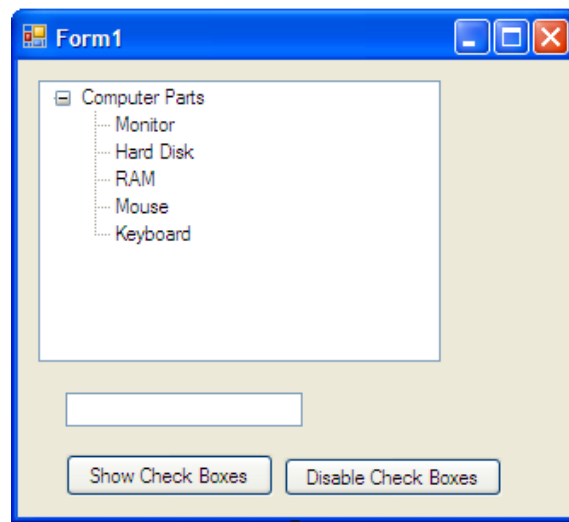
Step 3: Now run the program and click on any node then you look like the following:



Using Checkboxes in TreeView Control

TreeView Control is a popular Component to developing standard software. TreeView also can display checkboxes. You can make checkboxes appear in a tree view by setting the tree view's CheckBoxes property to True.

Step 1: Create a new project and make a windows form1, a TreeView Control, a Textbox and Two Buttons on windows form1. Now according to section 31.2 example create a TreeView Control with several nodes like as follows:



Step 2: Now double click on **Show Check Boxes** button and type the following code:

```
TreeView1.CheckBoxes = True
```

Step 3: Now double click on **Disable Check Boxes** button and type the following code:

```
TreeView1.CheckBoxes = False
```

Step 4: Now double click on **TreeView Control**, you will see the following code:

```
Private Sub TreeView1_AfterSelect(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles
TreeView1.AfterSelect

End Sub
```

Step 5: Now replace the above code segment with the following code:

TreeView1_AfterSelect replace with **TreeView1_AfterCheck**

TreeView1.AfterSelect replace with **TreeView1.AfterCheck**

Now the above code (step 4) will be as follows:

```
Private Sub TreeView1_AfterCheck(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles
TreeView1.AfterCheck

End Sub
```

Step 6: Now type the following code under TreeView Control:

```
If e.Node.Checked Then
    TextBox1.Text = "You Checked :" &
e.Node.Text
```

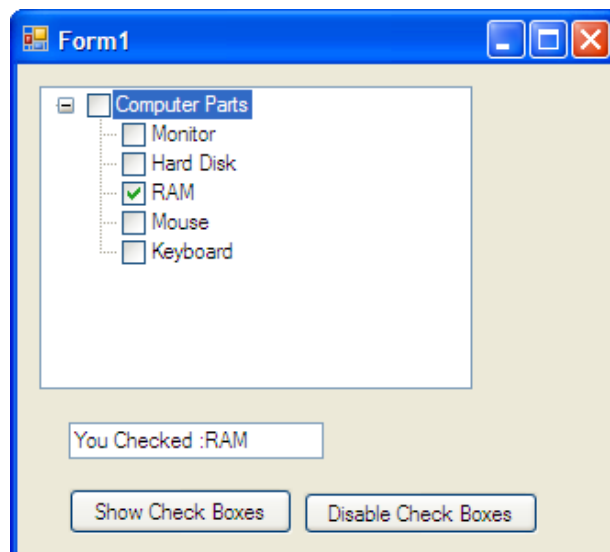


```
Else
    TextBox1.Text = "You UnChecked :" &
e.Node.Text
End If
```

Now full code will be as follows:

```
Private Sub TreeView1_AfterCheck(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles
TreeView1.AfterCheck
    If e.Node.Checked Then
        TextBox1.Text = "You Checked :" &
e.Node.Text
    Else
        TextBox1.Text = "You UnChecked :" &
e.Node.Text
    End If
End Sub
```

Step 7: Now run the program and click on **Show Check Boxes** button, you will see the checkboxes in front of node text and if you click on **Disable Check Boxes** button, the checkboxes will be hidden. If you click on check boxes of node like RAM, the corresponding text will be shown in textbox that is “You Checked:RAM ” and if you uncheck the checkboxes like RAM, the corresponding text will be shown in textbox tha is “ You UnChecked: RAM”which will look like the following:





Lesson 3.4-3.6

TabControl

Introduction

This lesson is described how to create a simple tab control in Visual Studio 2008. To make this possible, we use the TabControl which creates different views in the same page. As your application becomes crowded with various controls, you may find its form running out of space. To solve such a problem, you can create many controls on a form or container and display some of them only in response to some action from the user.

Upon completion of this unit you will be able to:



Outcomes

- Create to TabControl.
- Add/remove TabPage.

TabControl

TabControl is an effective component in Visual studio 2008, which is used to developing standard software. This is contains more than one tab pages. The TabControl component manages tab pages where each page may contains different child controls. You can create, add and remove controls using TabControl properties.

The Public properties of TabControl objects are listed below:

SL No.	Property Name	Description
1.	Alignment	This is used to gets/sets where the tabs appear(top, left, etc)
2.	Appearance	This is used to gets/sets the appearance of tabs in a tab control.
3.	DisplayRectangle	This is used to gets/sets the bounding rectangle of the tab pages
4.	HotTrack	This is used to gets/sets whether the tabs should change appearance when the mouse is over them.
5.	ImageList	This is used to gets/sets the images to show in



tabs.

- 6. Multiline This is used to gets/sets whether the tab control can show more than one row of tabs.
- 7. RowCount This is used to gets/sets the numbers of rows in the tab strip.
- 8. SelectedIndex This is used to gets/sets selected tab page's index.
- 9. ShowToolTips This is used to gets/sets whether a tab's tool tip can be displayed.
- 10. TabCount This is used to gets the number of tabs.
- 11. TabPages This is used to gets the collection of tab pages.


The public properties of TabPage objects are listed below:

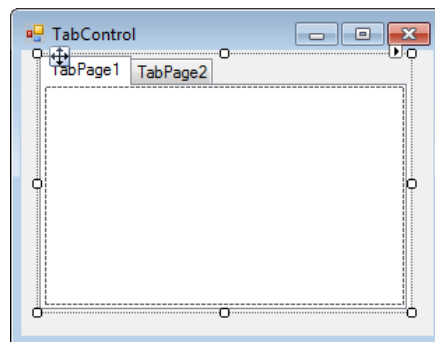
Sl.No	Property Name	Description
1.	ImageIndex	This is used to gets/sets the index of the image in this tab.
2.	Text	This is used to gets/sets the text to show in the tab.
3.	ToolTipText	This is used to gets/sets the tab's tool tip text.

Creating TabControls

After you have added a new tab control to a windows form at design time you can add tab pages to it by opening the TabPages property in the properties window.

Create a new Tab and tab pages you can follow the following steps:

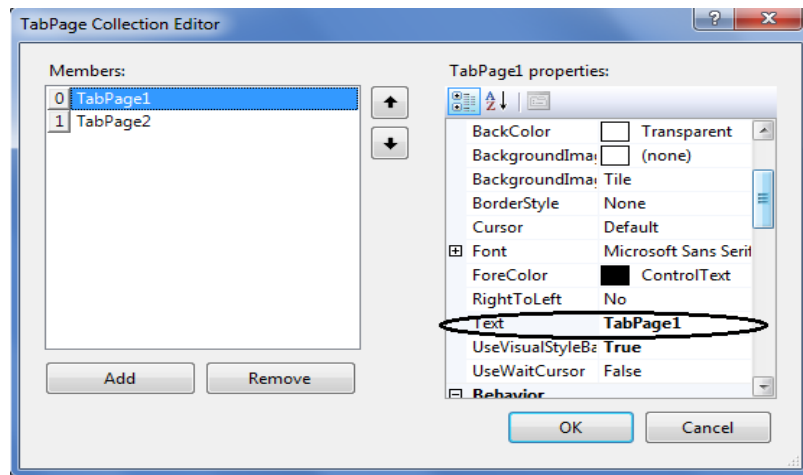
Step 1: First of all, create a windows form and then drag a TabControl  from toolbox and drop it on windows form, you will look like the following window.



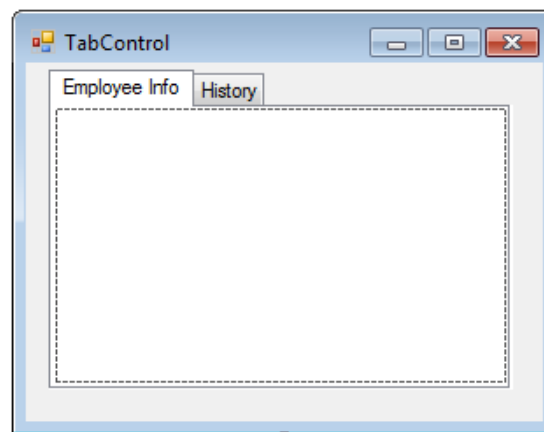
Step 2: Now as selected state, click on TabPages collection option from the properties window you will look TabPage collection editor, like the



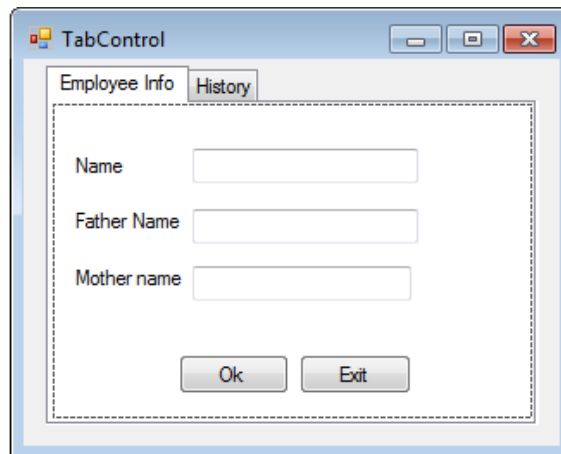
following:



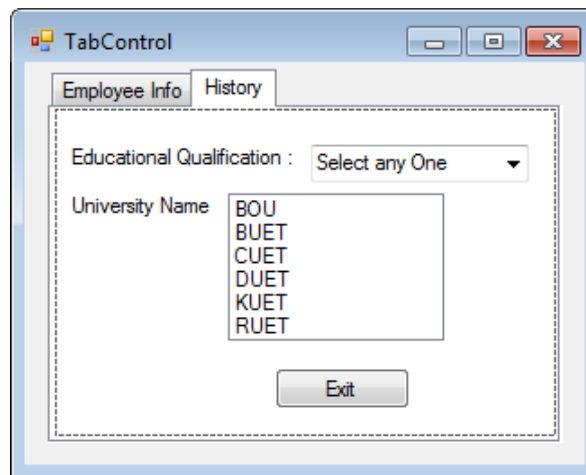
Step 3: Here you can add or remove Tab page from the members option by clicking Add or Remove button respectively. Now click on TabPage1, you can change the text of TabPage1 from the TabPage1 properties from the right side, which is marked as a black color circle. Now give the text name of TabPage1 as 'Employee info' from the TabPage1 properties window and now click on TabPage2 from left side members option and give the text name of TabPage2 as 'History ' from the TabPage2 properties and then click ok button. If you want to add another TabPage, just click add button and change the corresponding properties. If you want to remove any TabPage, Just click on that TabPage and click remove button. After completing this work you will look like the following window:



Step 4: you can create more than one another component on the any tab page. Now create three Levels, three TextBoxes and two Buttons on Employee Info Tab page, and give the name as you want. Here we shows an example page which will look like the following:



Step 5: Now create Two Levels, One ComboBox and One ListBox and one Button on History Tab page, and give the name as you want. Here we shows an example page which will look like the following:



Step 6: Now double click on exit button and type the code 'End' under exit button of both tab pages, which will look like the following:

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
```

```
End
```

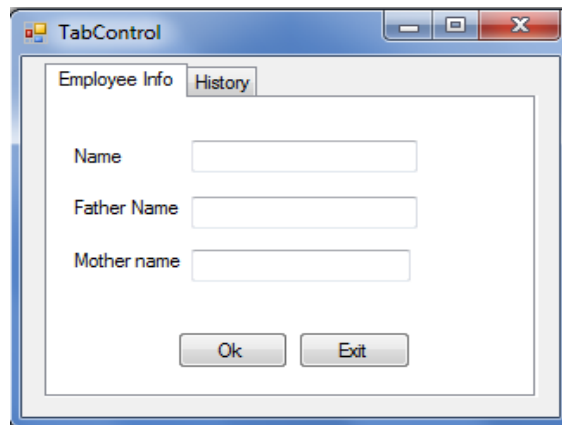
```
End Sub
```

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
```

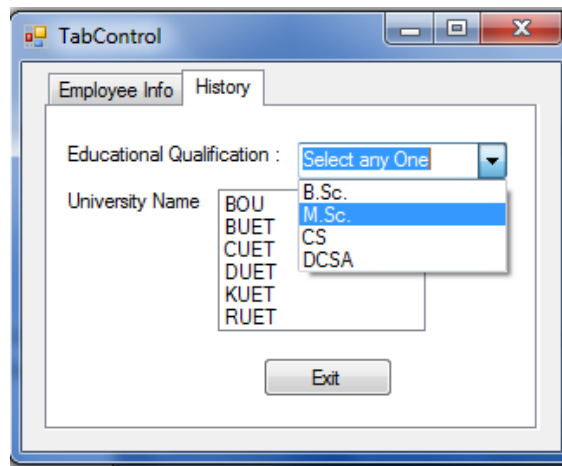
```
End
```

```
End Sub
```

Step 7: Now run the program by clicking on run button from the tools bar or press F5 key from the keyboard. You will look like the following:



Step 8: Now click on History Tab and click select educational qualification from combobox, you will look like the following:



Step 9: Now if you want to exit the program just click on exit button.

Assessment



Assessment

Exercise

1. Write the steps to create TabControl.



Lesson 3.7

TrackBar Component

Upon completion of this unit you will be able to:



Outcomes

- *Know* TrackBar control.
- *Handle* TrackBar Events.
- *Setting* TrackBar Ticks.

Basic about TrackBar

TrackBar control works much like scroll bar, but they have a different appearance, resembling the controls you'd find on stereo. This control shows on various audio software. TrackBar also can display ticks, giving the user an idea of the scale used to set the controls value.

The public properties of TrackBar object is shown in following table:


SI No.	Name of Property	Description
1.	AutoSize	This property is used to gets/sets if the TrackBar's height or width should be automatically sized.
2.	ForeColor	This property is used to holds the foreground color of the TrackBar.
3.	LargeChange	This property is used to gets/sets the value added to or subtract from to the value property when the scroll box moves a large distance.
4.	Maximum	This property is used to holds the upper limit of the range of the TrackBar
5.	Minimum	This property is used to holds the lower limit of the range of the TrackBar
6.	Orientation	This property is used to gets/sets the horizontal or vertical orientation of the TrackBar.
7.	SmallChange	This property is used to gets/sets a value which is added to or subtracted from the value property when the scroll box moves a small distance.

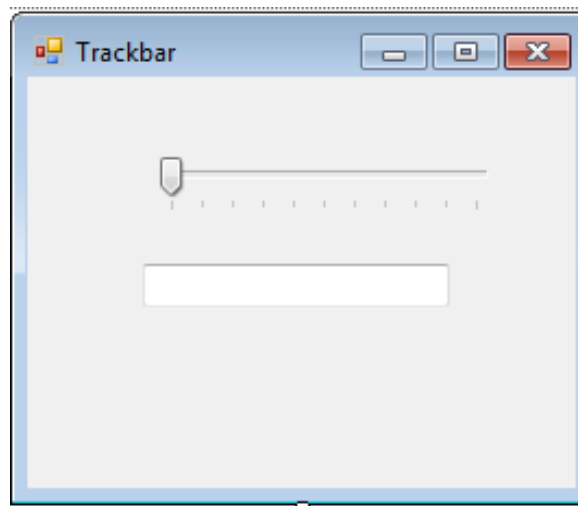


- | | | |
|-----|---------------|--|
| 8. | TickFrequency | This property is used to gets/sets a value specifying the distance between ticks. |
| 9. | TickStyle | This property is used to gets/sets how to display the tick marks in the TrackBar. |
| 10. | Value | This property is used to gets/sets the current position of the slider in the trackBar. |

Handling TrackBar Events

TrackBar have two events such as Scroll event and ValueChanged event. You can get the current value of the trackBar with the Value property. The procedure to create a TrackBar control is shown in below step by step.

Step 1: First of all create a windows form and then drag the TrackBar icon  from the Toolbox and drop it on windows form and also create a Textbox on windows form. You will look like the following window:

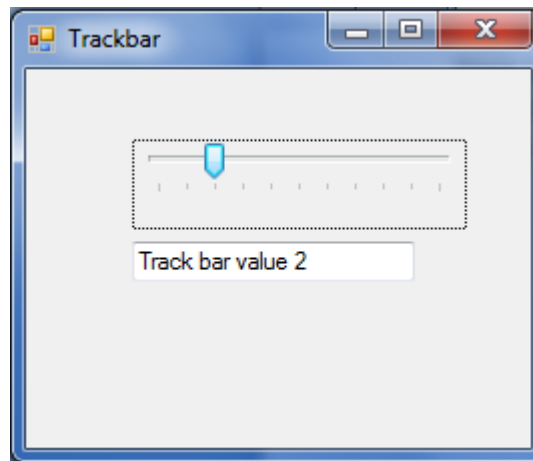


Step 2: Here by default you will look maximum 10 tick. The space between two vertical line of TrackBar is called tick. If you want to change this value you can change the maximum and minimum properties of TrackBar from properties window.

Step 3: Now double click on TrackBar control from the windows form and then type the following code:

```
TextBox1.Text = "Track bar value " & TrackBar1.Value
```

Step 4: Now Run the program and change the Vertical bar of TrackBar you will look like the following window:



If you want to configure TrackBar control, you can use the TickStyle property, which lets you determine how ticks are displayed. This property can take values from the TickStyle enumeration. You will look this enumeration from the TrackBar Properties window. Enumerations are as follows:

Both: Tick marks are located on both sides of the control

BottomRight: Tick marks are located on the bottom of a horizontal control or on the right side of a vertical control.

None: No tick marks appear in the TrackBar control.

ToLeft: Tick Marks are located on the top of a horizontal control or on the left of a vertical control.

You can set the tick frequency, which sets the distance between ticks with the TickFrequency property.

Assessment



Assessment

Exercise

1. Write short notes on TrackBar events.



Lesson 3.8

Timer

Introduction

In this lesson, we shall show you how to use timer in VB2008. Timer is a useful control in Visual Basic. Timer is used to control and manage events that are time related. For example, you need timer to create a clock, a stop watch, a dice, animation and more. Timer is a hidden control at runtime, like the engine of an automobile. We shall illustrate the usage of timer using a few examples

Upon completion of this unit you will be able to:



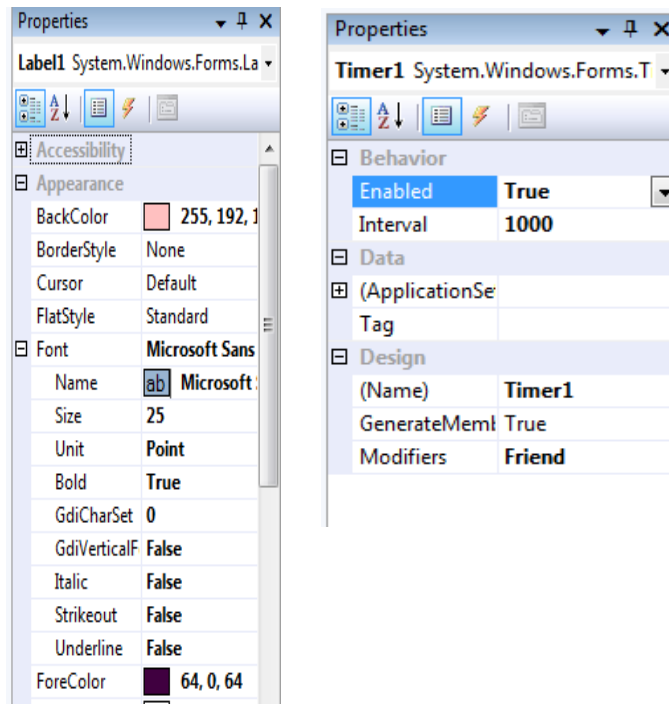
Outcomes

- Use timer.

Creating Digital Clock

In order to create a clock, you need to use the Timer control that comes with Visual Basic 2008. The Timer control is a control object that is only used by the developer, it is invisible during runtime and it does not allow the user to interact with it.

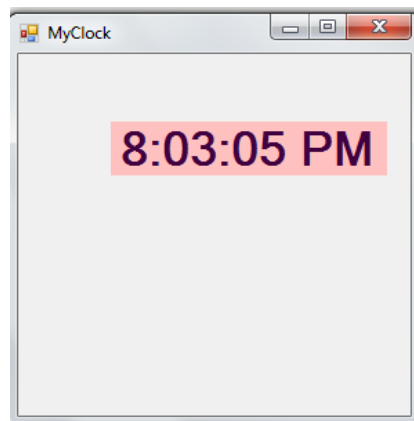
To create the clock, first of all start a new project in Visual Basic 2008 Express and select a new Windows Application. You can give the project any name you wish, but we will name it MyClock. Change the caption of the Form1 to MyClock in the properties window. Now add the Timer control to the form by dragging it from the ToolBox. Next, insert a label control into the form. Change the Font size of the label to 25 or any size you wish, set the Font alignment to be middle center and change the ForeColor and BackColor. Before we forget, you shall also set the Interval property of the Timer control to 1000, which reflects a one second interval. You also need to ensure that the Enabled property of the Timer control is set to true so that the clock starts running as soon as it is loaded.



Now, you are ready for the coding.

```
Private Sub Timer1_Tick(ByVal sender As System.Object,  
ByVal e As System.EventArgs) Handles Timer1.Tick  
    Label1.Text = TimeOfDay  
End Sub
```

The digital clock is as shown in the following Figure





Creating Stopwatch

We can create a simple stopwatch using the Timer control. Start a new project and name it StopWatch. Change the Form1 caption to StopWatch. Insert the Timer control into the form and set its interval to 1000 which is equal to one second. Also set the timer Enabled property to False so that it will not start ticking when the program is started. Insert three command buttons and change their names to StartButton, StopButton and ResetButton respectively. Change their text to “Start”, “Stop” and “Reset” accordingly. Now, write the code as follows:

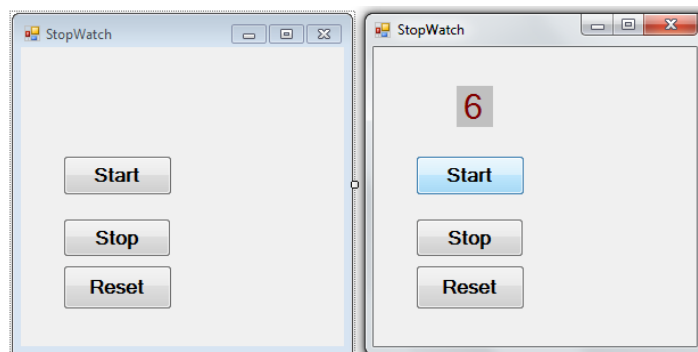
```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
    Label1.Text = Val(Label1.Text) + 1
End Sub

Private Sub StartButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles StartButton.Click
    Timer1.Enabled = True
End Sub

Private Sub StopButton (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    Timer1.Enabled = False
End Sub

Private Sub ResetButton (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Label1.Text = 0
End Sub
```

The Interface of the Stopwatch is as shown below:





Lesson 3.9 – 3.10

Image

Introduction

PictureBox controls are among the most powerful and complex items in the Visual Basic Toolbox window. Once you place a PictureBox on a form, you might want to load an image in it, which you do by setting the *Picture* property in the Properties window. You can load images in many different graphic formats, including bitmaps (BMP), device independent bitmaps (DIB), metafiles (WMF), enhanced metafiles (EMF), GIF and JPEG compressed files, and icons. You can decide whether a control should display a border, resetting the *BorderStyle* to 0-None if necessary. Another property that comes handy in this phase is *AutoSize*: Set it to True and let the control automatically resize itself to fit the assigned image.

Upon completion of this unit you will be able to:

- Use picture onto Form.



Outcomes

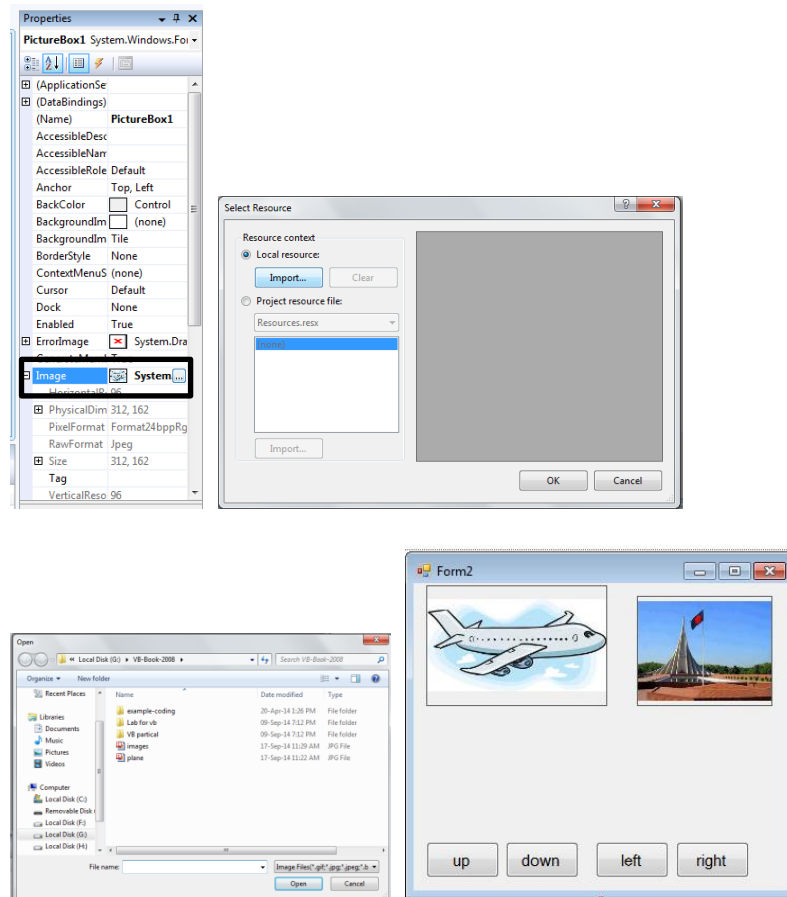
PictureBox

PictureBox control let you display an image, so let's try it out start a new VB **Windows** Forms project and drag-and-drop a PictureBox onto the form. The PictureBox has a range of standard properties and methods and some that allow you to load a graphics file and control how it is displayed.

Image Property

The most important property of either type of control is its Image property which specifies the graphics file that will be displayed. Set the Image property in the Properties window the Select Resource **dialog box** appears. This lets you specify two different way of working with files. You can select Project resource file or you can select Local resource. The difference is that if you select Project resource file the image file is copied into the project and it gets distributed with the application automatically. If you select Local resource then the file is left where you put it and it is just

used by the program. Of course this means that if you distribute the program to other people you have to make sure that you include a copy of the file and make sure it is stored in the correct location. You can display a file in GIF, BMP, JPEG, PNG or TIFF format.

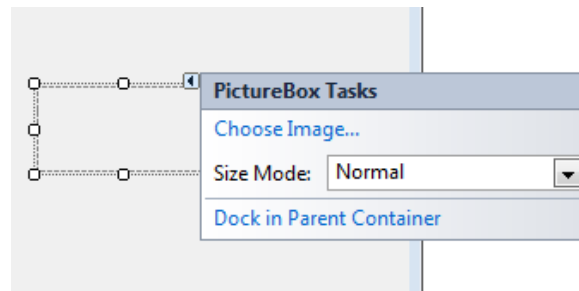


You can do more interesting things at run time. To begin with, you can programmatically load any image in the control by using

```
Private Sub Form2_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
    PictureBox3.ImageLocation = ("G:\VB-Book-
2008\shahidminar.jpg")
End Sub
```

Controlling the image

There are a small number of properties that you can set to control how the image is displayed in the PictureBox. The most important is the Size Mode property.



If you set this to Normal then the image will be displayed in the PictureBox at its correct size. If the picture box is too small then you only see what fits into the display area and if it is too big the image is surrounded by a Background fill.

If you set this to StretchImage the graphic is scaled so that it fills the current size of the PictureBox.

Zoom works in the same way as StretchImage but the scaling doesn't distort the image i.e. it keeps the **aspect ratio** fixed. Of course this means that the image might not fill the PictureBox in one dimension.

If you set this to AutoSize then it is the PictureBox which changes its size to always fit the size of the image being displayed.

CenterImage works like Normal but the image is centered rather than being in the top left hand corner.

Creating Animation

Although Visual Basic 2008 is generally a programming language, it can also be used to create animation. In this section, we will show you how to move an object by pressing a command button. You need to make use of the Top and Left properties of an object to create animation. The Top property defines the distance of the object from the top most border of the screen while the Left property defines the distance of the object from leftmost border of the screen. By adding or subtracting the distance of the object we can create the animated effect of moving an object.

Start a new project and name it as animation, or any name you wish. Now In the PictureBox properties window, select the image property and click to import an image file from your external sources such as your hard drive, your Pendrive or DVD. Next, insert command buttons; change their captions to down, left.

Now,

Click on the buttons and key in the following code:

```
Private Sub MoveDownBtn_Click(ByVal sender As System.Object,
```




```
ByVal e As System.EventArgs) Handles MoveDownBtn.Click
```

```
PictureBox1.Top = PictureBox1.Top + 10
```

```
End Sub
```

```
Private Sub MoveLeftBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MoveLeftBtn.Click
```

```
PictureBox1.Left = PictureBox1.Left - 10
```

```
End Sub
```

```
Private Sub MoveRightBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MoveRightBtn.Click
```

```
PictureBox1.Left = PictureBox1.Left + 10
```

```
End Sub
```

Explanation:

Each time the user clicks on the Move Down button, the distance of the PictureBox increases by 10 pixels from the top border, creating a downward motion. On the other hand, each time the user clicks on the Move Up button, the distance of the PictureBox decreases by 10 pixels from the top borders, thus creating an upward motion.

Creating Animation using Timer

We can create continuous animation using timer without the need to manually clicking a command button. We can create left to right or top to bottom motion by writing the necessary code.

First of all, insert a PictureBox into the form. In the PictureBox properties window, select the image property and click to import an image file from your external sources such as your hard drive, your Pendrive or DVD. Next, insert a Timer control into the form set its interval property to 100, which is equivalent to 0.1 second. Finally, add two command button to the form, name one of them as AnimateBtn and the other one as StopBtn, and change to caption to Animate and Stop respectively.

We make use of the Left property of the PictureBox to create the motion. PictureBox.Left means the distance of the PictureBox from the left border of the Form . Now click on the Timer control and type in the following code:

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick  
If PictureBox1.Left < Me.Width Then  
PictureBox1.Left = PictureBox1.Left + 10  
Else  
PictureBox1.Left = 0
```



```
End If  
End Sub
```

In the code above, `Me.Width` represents the width of the Form. If the distance of the `PictureBox` from the left is less than the width of the Form, a value of 10 is added to the distance of the `PictureBox` from the left border each time the Timer tick, or every 0.1 second in this example. When the distance of the `PictureBox` from the left border is equal to the width of the form, the distance from the left border is set to 0, which move the `PictureBox` object to the left border and then move left again, thus creates an oscillating motion from left to right. We need to insert a button to stop motion. The code is:

```
Timer1.Enabled = False
```

To animate the `PictureBox` object, we insert a command button and key in the following code:

```
Timer1.Enabled = True
```

The Image of the Animation program is shown below: